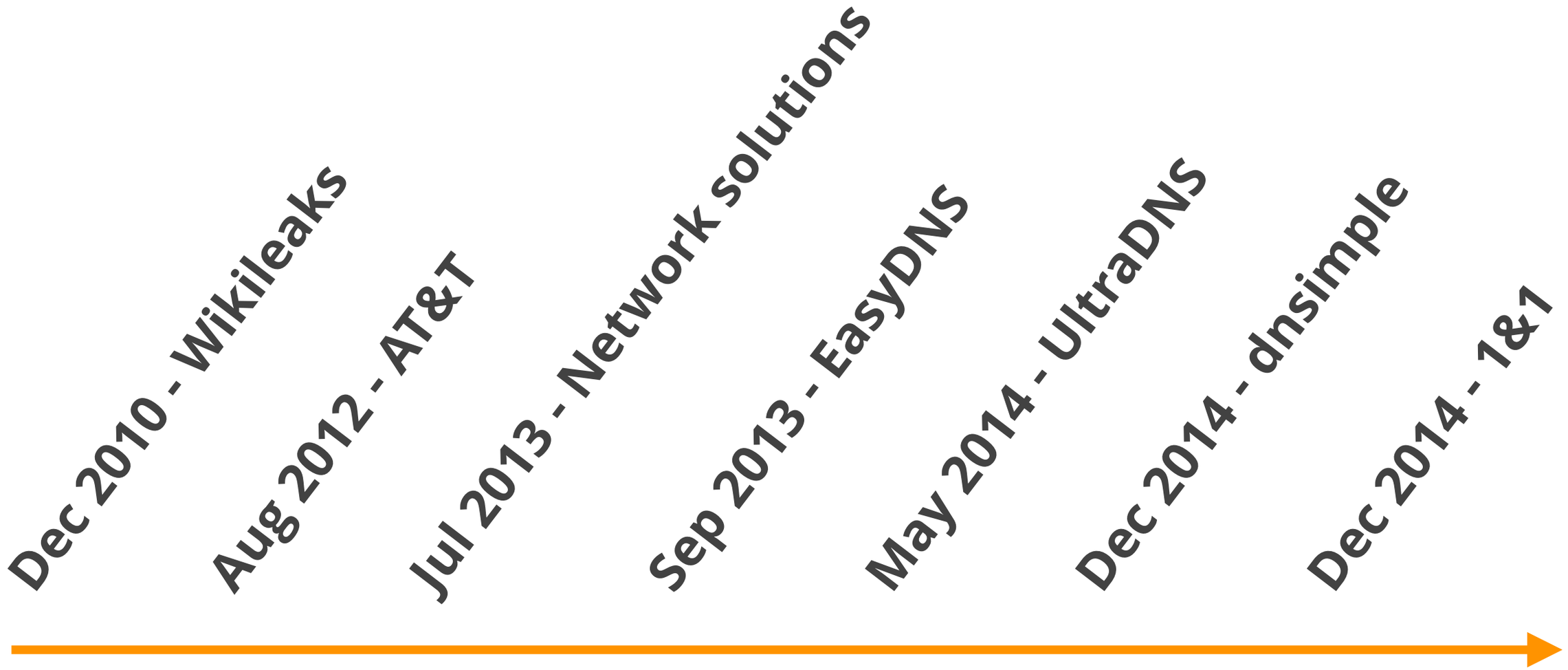
Dealing with DNS packet floods

Marek Majkowski

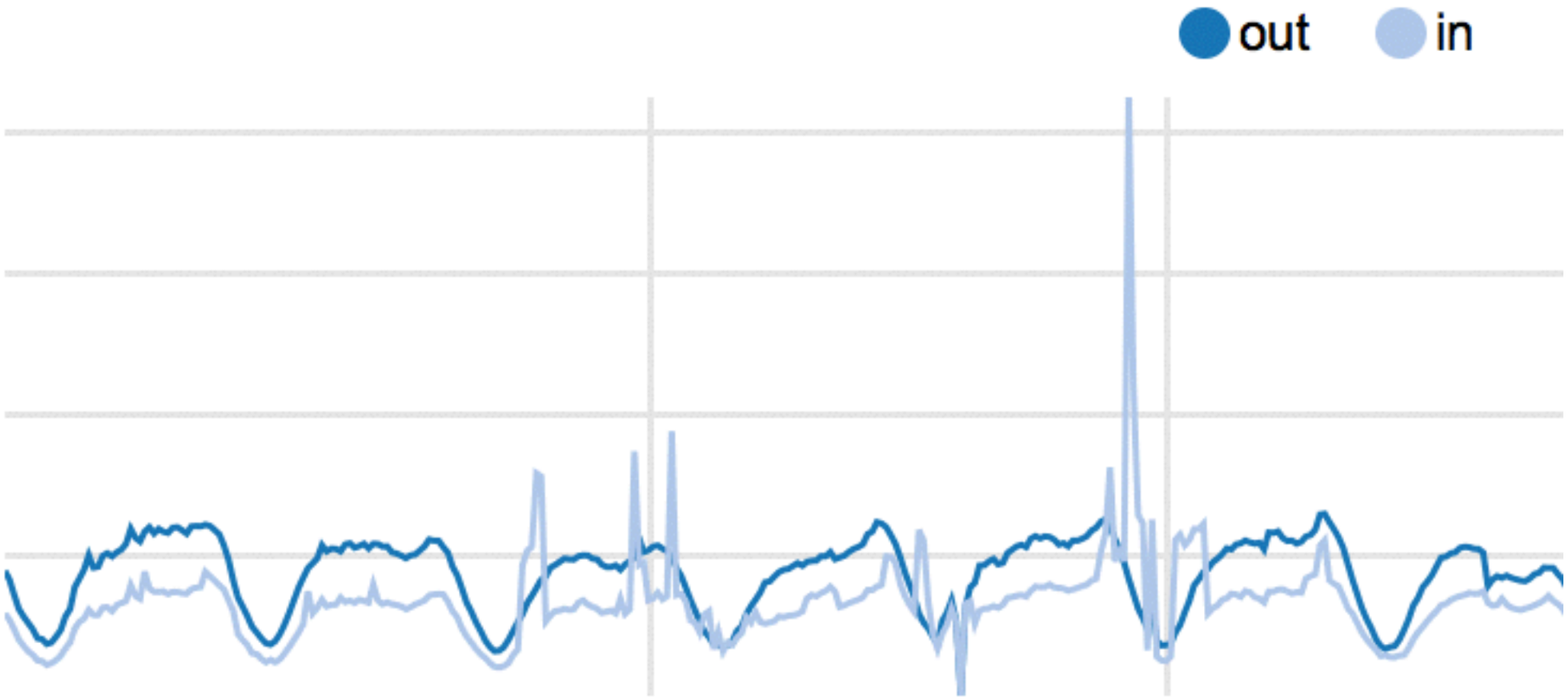
1. Network mitigations
2. All about dropping
3. Automation

Everyone gets flooded



Usual traffic

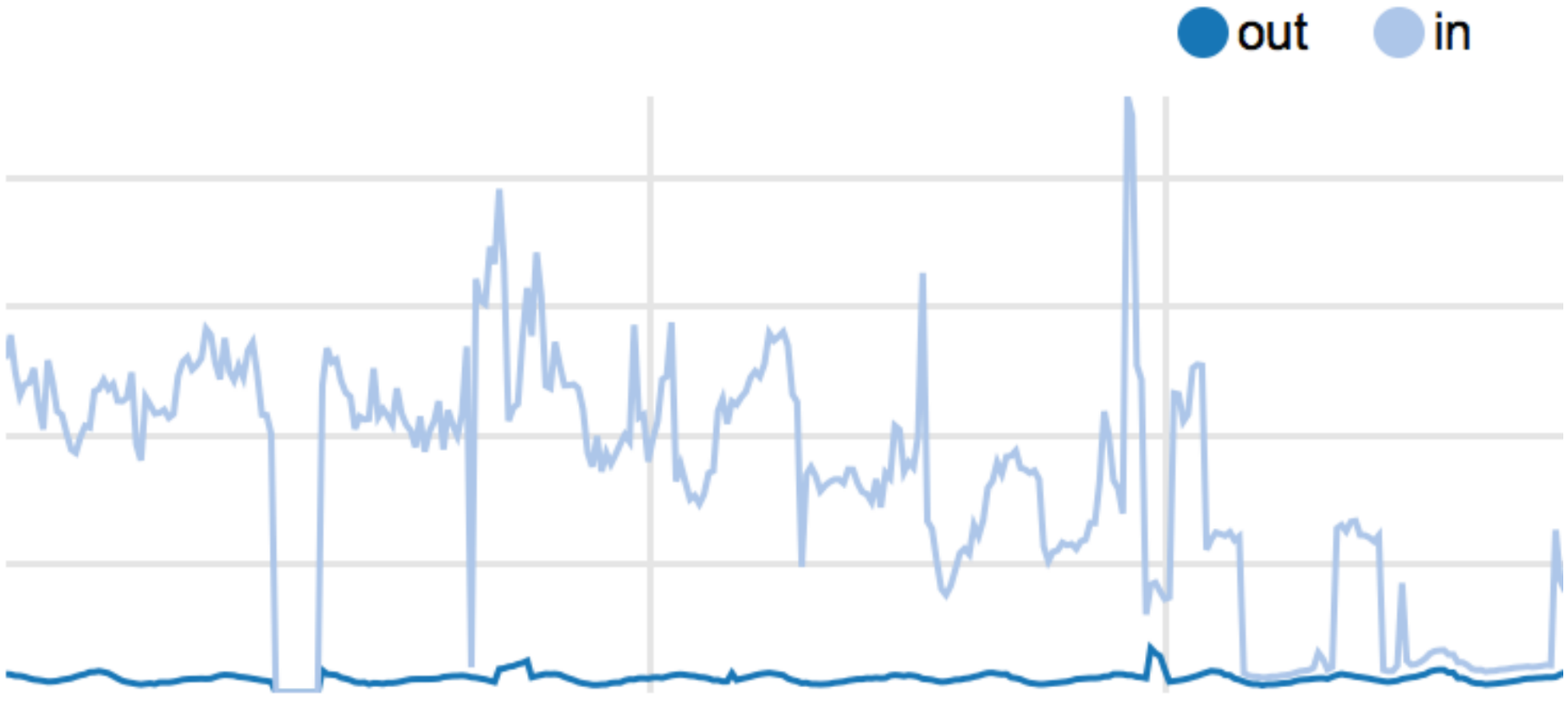
pps



7 days →

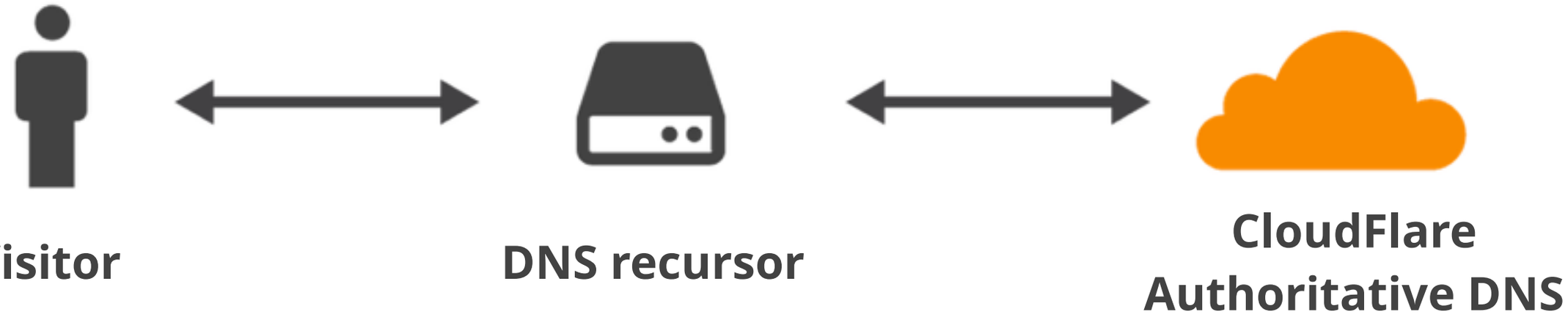
Flood traffic

pps



7 days →

CF as authoritative DNS



What hits us

```
$ dig example.com NS

;; QUESTION SECTION:
;example.com.          IN  NS

;; ANSWER SECTION:
example.com. 21599   IN  NS  paul.ns.cloudflare.com.
example.com. 21599   IN  NS  emma.ns.cloudflare.com.
```

- DNS requests (pps)
- SYN floods (bps)
- Hit and run (TR / SLIP may not work)

Chapter 1

Network mitigation

Let’s talk about the scale

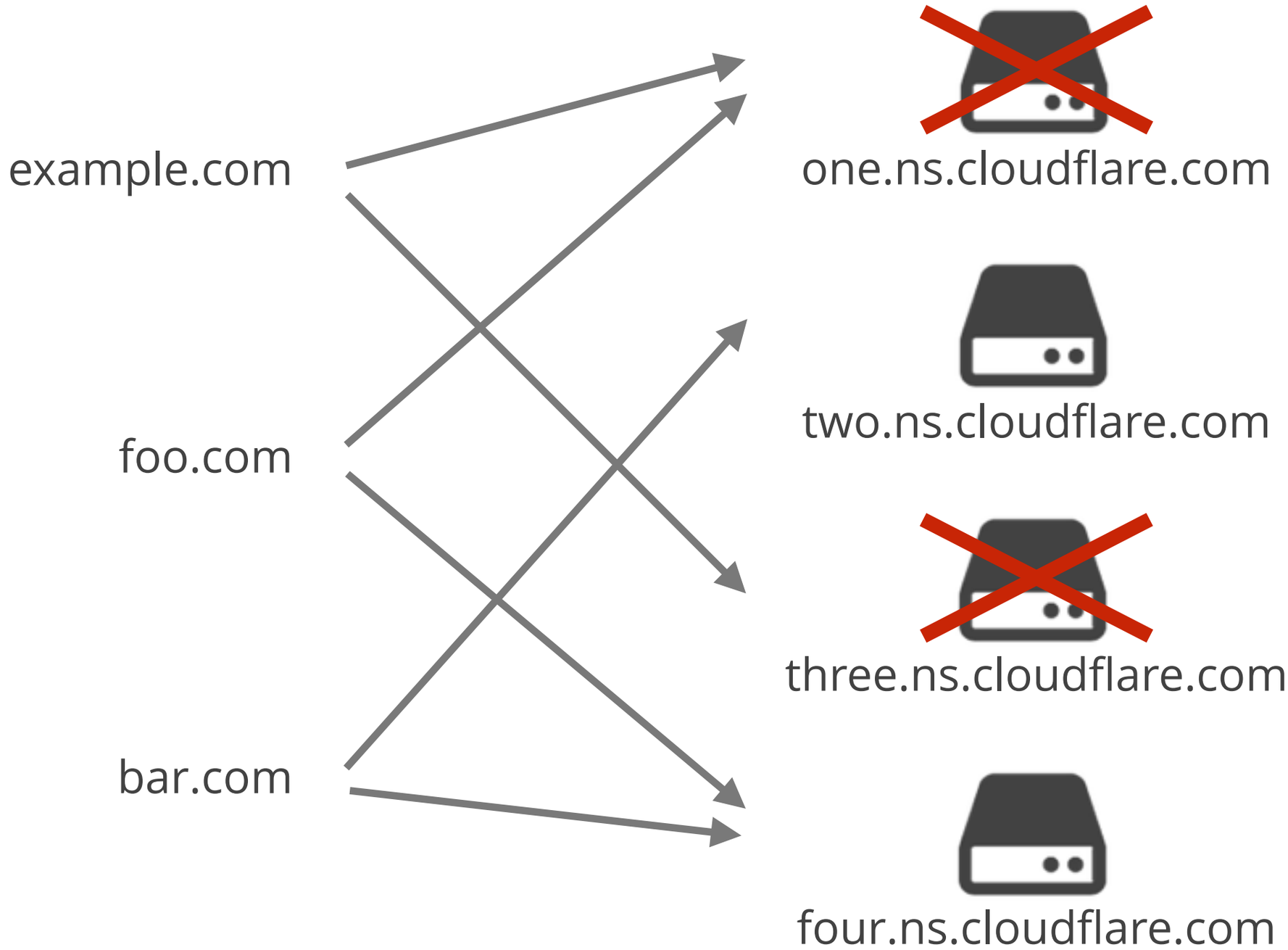
	congestion		
	10M pps		
	6M pps		
	1.2M pps		
	0.3M pps		

upstream: capacity game

upstream	congestion	more ports, null, topology	ip
	10M pps		
	6M pps		
	1.2M pps		
	0.3M pps		

Topology: anycast

Topology: handle the null



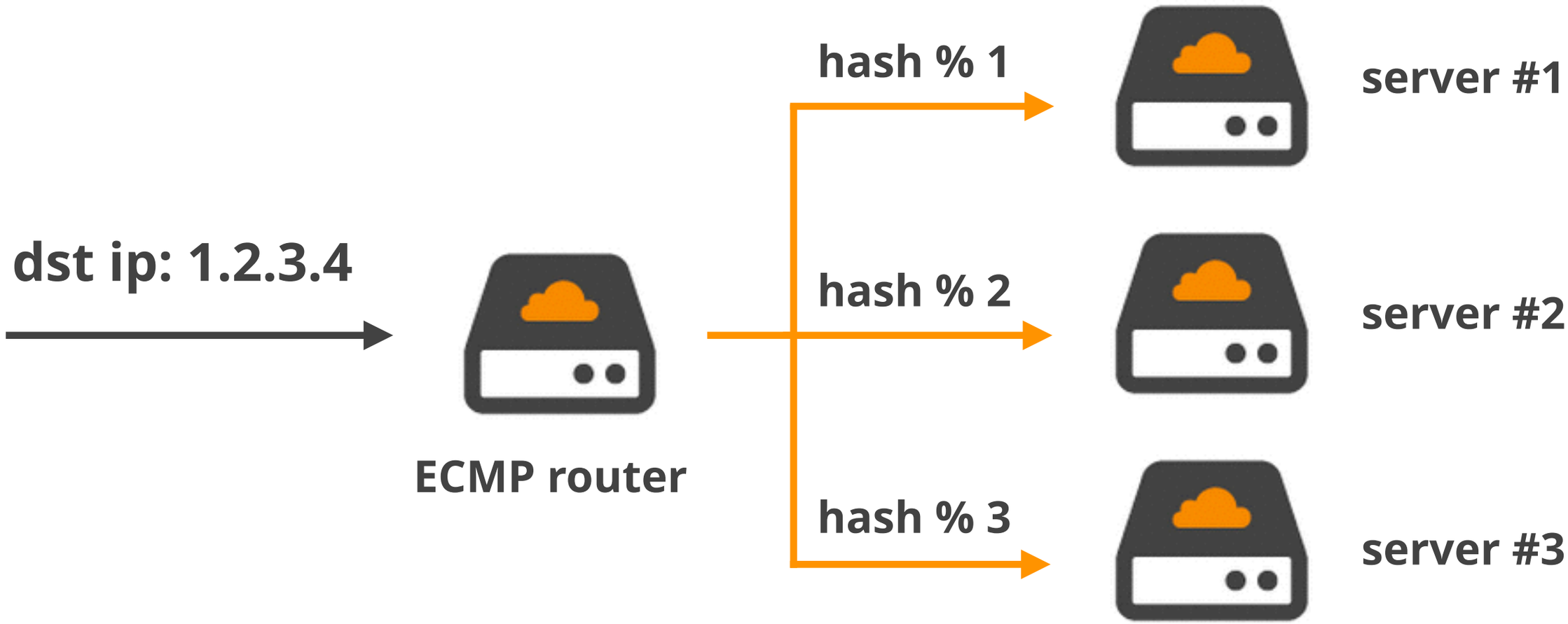
New trend

- “foo01.com”, “foo02.com”, “foo03.com”
- Flood against all domains start at the same time
- Beware of allocation of name servers

Scale: router

upstream	congestion	more ports, null, topology	ip
router	**10M pps**	**ECMP, flowspec**	**ip,proto, length**
	6M pps		
	1.2M pps		
	0.3M pps		

ECMP: spread it out



Flowspec

```
+     route 173.245.59.133-UDP-DISCARD {
+         match {
+             destination 173.245.59.133/32;
+             protocol 17;
+             port 53;
+             packet-length 1500;
+         }
+         then discard;
+     }
```

- router-based, centrally managed firewall
- uses BGP as transport
- patchy vendor support, patchy ipv6 support
- coarse grained, can’t inspect payload

Scale: DNS server

upstream	congestion	more ports, null, topology	ip
router	10M pps	ECMP, flowspec	ip, proto, length,
	6M pps		
	1.2M pps		
DNS server	**0.3M pps**	**selective drops, just handle**	**full payload**

DNS server

- Linux network stack is “slow” (??k pps per core)
- No point in dropping - most of the work is to receive and parse the packet
- We had rules, but weren’t too effective
- Bind to specific IPs 1.2.3.4:53, not to 0.0.0.0:53
- (RRLs is another subject)

Scale: Iptables traditional

upstream	congestion	more ports, null, topology	ip
router	10M pps	ECMP, flowspec	ip, proto, length,
	6M pps		
kernel	**1.2M pps**	**iptables traditional**	**ip,proto, length, fixed offset bits**
DNS server	0.3M pps	selective drops, just handle	full payload

Iptables u32

- u32 module is well known
- Hard to use and error prone
- Well documented to use in DNS

```
iptables -m u32 –u32 \
  ”6&0xFF=0x6 && 4&0x1FFF=0 && 0>>22&0x3C@4=0x29”
```

Iptables BPF

- BPF is better, more generic
- Does fairly complex, yet fast matching

```
iptables -A INPUT \
    -p udp --dport 53 \
    -m bpf --bytecode "14,0 0 0 20,177 0 0 0,12 0 0 0,7 0 0 0,64 0 0 0,\
                       21 0 7 124090465,64 0 0 4,21 0 5 1836084325, \
                       64 0 0 8,21 0 3 56848237,80 0 0 12,21 0 1 0, \
                       6 0 0 1,6 0 0 0," \
    -j DROP
```

Scale: Iptables BPF

upstream	congestion	more ports, null, topology	ip
router	10M pps	ECMP, flowspec	ip, proto, length,
	6M pps		
kernel	**1.2M pps**	**iptables bpf**	**full payload**
DNS server	0.3M pps	selective drops, just handle	full payload

Chapter 2

Why dropping in BPF works

Tcpdump expressions

- Originally: `tcpdump -n “udp and port 53”`
- Now: cls_bpf, seccomp-bpf, etc
- xt_bpf implemented in 2013 by Willem de Bruijn
- Need to deal with BPF byte code
- Tools around it are scarce (tcpdump expressions)

```
./iptables -A OUTPUT -m bpf --bytecode '6,40 0 0 14, 21 0 3 2048,48 0 0 \
    0 1 20,6 0 0 96,6 0 0 0,' -j

(as generated by tcpdump -i any -ddd ip proto 20 | tr '\n' ',')
```

```
$ ./bpfgen -o 14 -s dns -- *.example.com
    ldx 4*([14]&0xf)
    ; l3_off(14) + 8 of udp + 12 of dns
    ld #34
    add x
    tax
    ; a = x = M[0] = offset of first dns query byte
    ; st M[0]

lb_0:
    ; ldx M[0]
    ; Match: *
    ldb [x + 0]
    add x
    add #1
    tax
    ; Match: 076578616d706c6503636f6d00 '\x07example\x03com\x00'  mask=00000000000000000000000000
    ld [x + 0]
    jneq #0x07657861, lb_1
    ld [x + 4]
    jneq #0x6d706c65, lb_1
    ld [x + 8]
    jneq #0x03636f6d, lb_1
    ldb [x + 12]
    jneq #0x00, lb_1
    ret #1

lb_1:
    ret #0
```

```
$ ./bpfgen -o 14 dns -- *.example.com
18,177 0 0 14,0 0 0 34,12 0 0 0,7 0 0 0,80 0 0 0,12 0 0 0,4 0 0 1,7 0 0 0,64 0 0 0,21 0 7
124090465,64 0 0 4,21 0 5 1836084325,64 0 0 8,21 0 3 56848237,80 0 0 12,21 0 1 0,6 0 0 1,6 0 0 0,
```

BPF bytecode

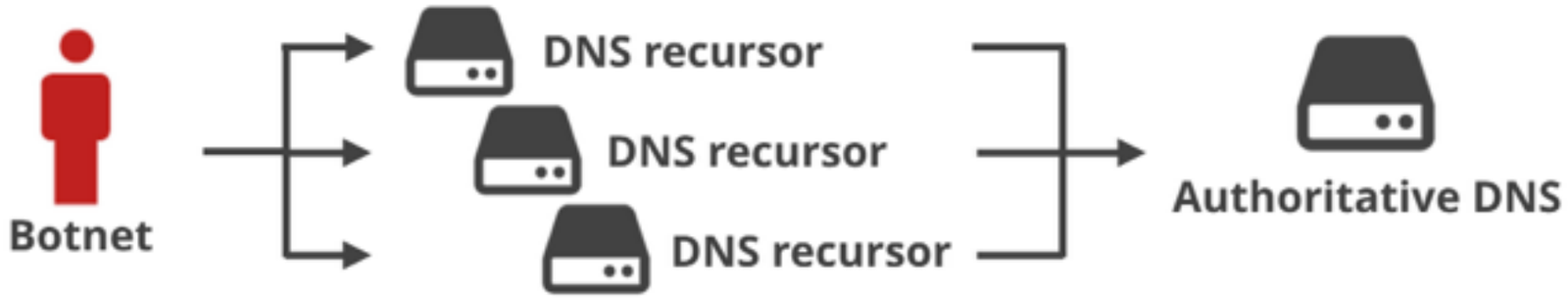
- Open source:
  - https://github.com/cloudflare/bpftools
- Can match various patterns:
  - *.example.com
  - ??.example.com
  - *{1-4}.example.com
  - —case-insensitive *.example.com
  - —invalid-dns

Just DROP.

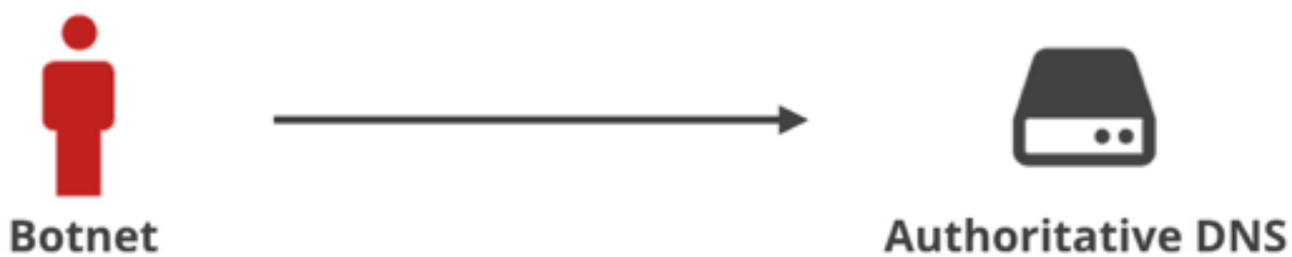
What hits AUTH

- Valid traffic

- Indirect floods, using recursors



- Direct floods, spoofing source IP

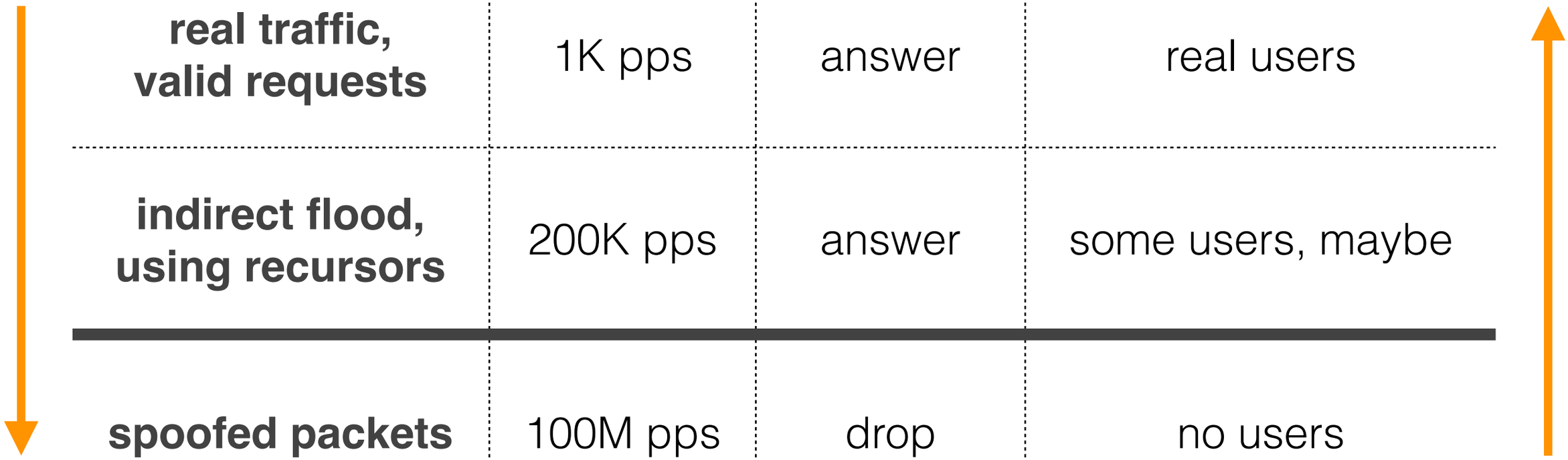


What should AUTH do

traffic category	scale	perfect action	
real traffic, valid requests	1K pps	answer	
indirect flood, using recursors	200K pps	answer	
spoofed packets	100M pps	drop	

What should AUTH do

traffic category	scale	perfect action	
real traffic, valid requests	1K pps	answer	real users
indirect flood, using recursors	200K pps	answer	some users, maybe
spoofed packets	100M pps	drop	no users



Spot fake packets

- “your heart condition?.foo.com”
- “www.foo.com,foo.com”
- “http://foo.com”
- “ubhcbattr.foo.qdedezsbm.gov.foo”
- “www.foo.com”
- “avhiwhun.www.foo.com”
- “xtnqafzfb.foo.com”

Spot fake packets

- “your heart condition?.foo.com” ← **spoofed**
- “www.foo.com,foo.com” ← **spoofed**
- “http://foo.com” ← **spoofed**
- “ubhcbattr.foo.qdedezsbm.gov.foo” ← **likely spoofed**
- “www.foo.com” ← **99% spoofed**
- “avhiwhun.www.foo.com” ← **may be real**
- “xtnqafzfb.foo.com” ← **may be real**

More selectors

- Anycast helps
  - Blacklisting non-regional IPs
  - Whitelisting valid recursor IPs
- Unusual EDNS
- Correlation in IP TTL
- Correlation in IP ID
- Unusual upper/lower case

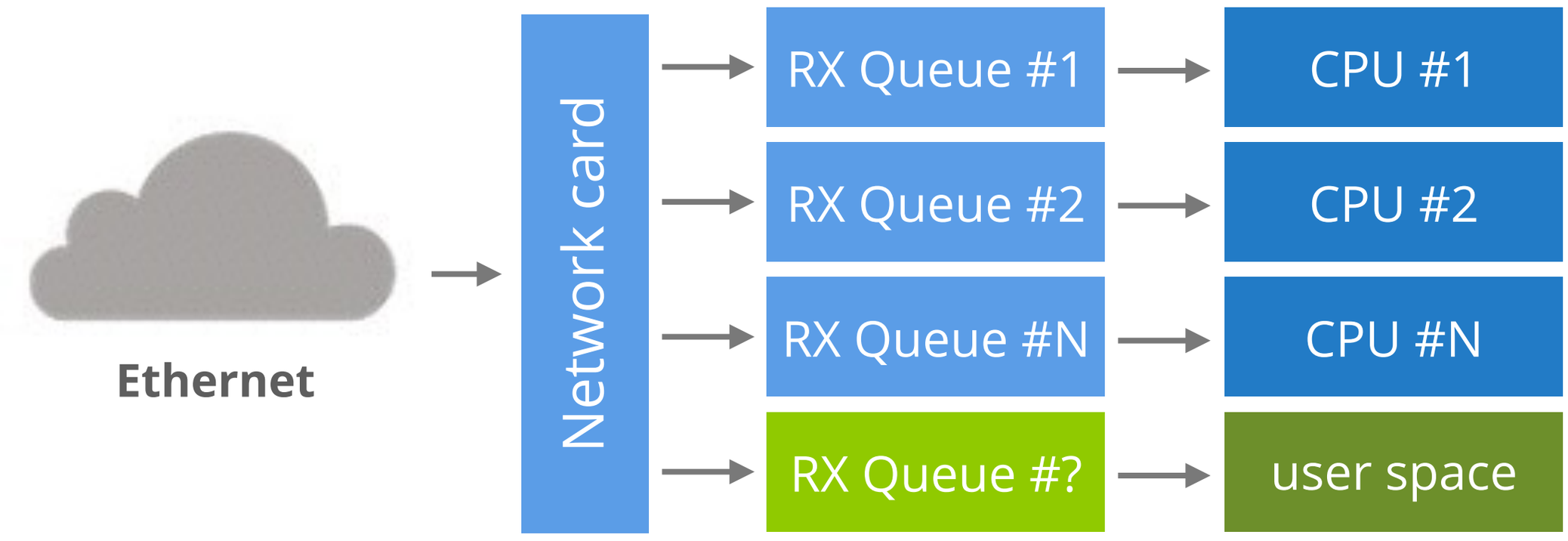
Managing the impact

traffic category	scale	perfect action	*.example.com - whitelist (ratelimited)	*.example.com - whitelist	*.example.com
real traffic, valid requests	1K pps	answer	answer	answer	drop
indirect flood, using recursors	200K pps	answer	some dropped	drop	drop
spoofed packets	100M pps	drop	drop	drop	drop

Scale: Iptables is slow

upstream	congestion	more ports, null, topology	ip
router	10M pps	ECMP, flowspec	ip, proto, length,
	6M pps		
kernel	**1.2M pps**	**iptables bpf**	**full payload**
DNS server	0.3M pps	selective drops, just handle	full payload

Floodgate


Ethernet
Network card
RX Queue #1
RX Queue #2
RX Queue #N
RX Queue #?
CPU #1
CPU #2
CPU #N
user space

Scale: Floodgate

upstream	congestion	more ports, null, topology	ip
router	10M pps	flowspec	ip, proto, length,
network card	**6M pps**	**floodgate**	**full payload**
kernel	1.2M pps	iptables	full payload
DNS server	0.3M pps	selective drops, just handle	full payload

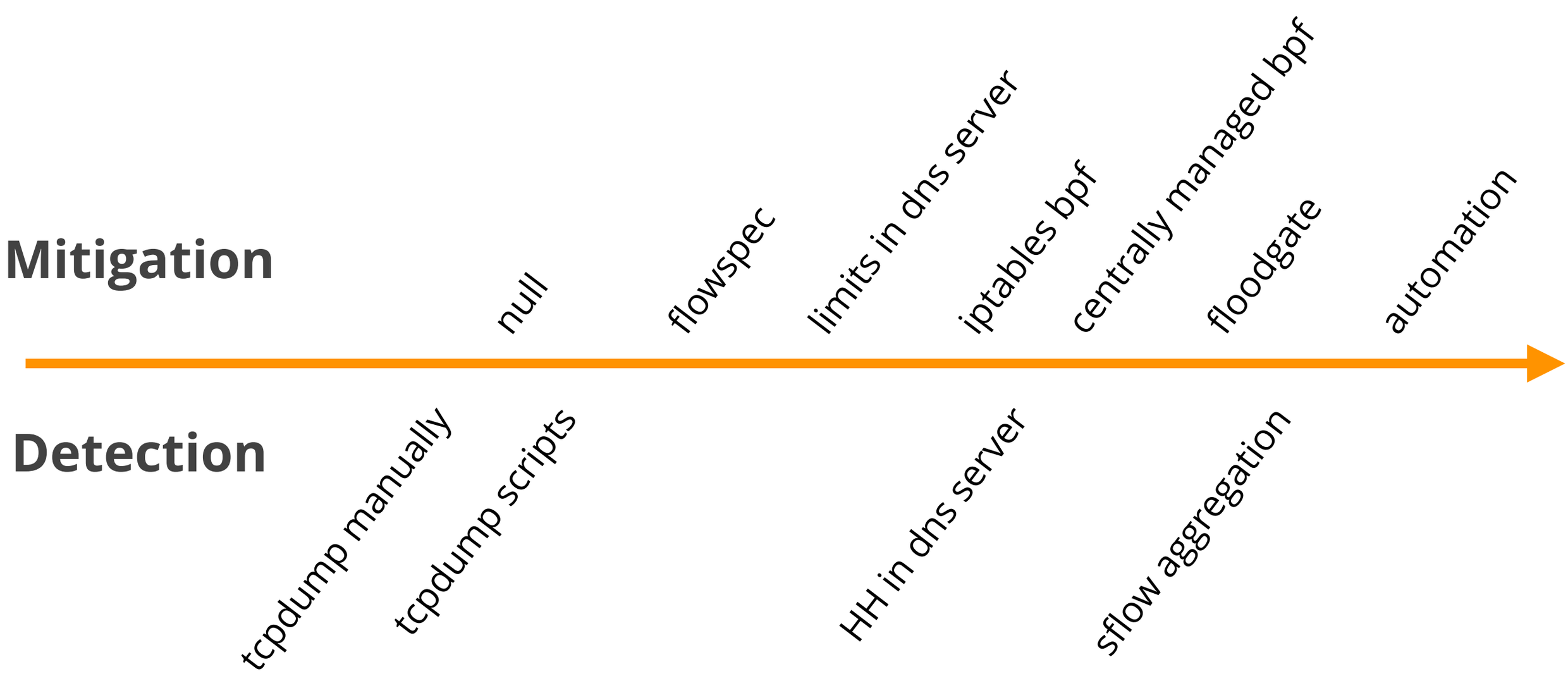
Chapter 3

Mitigation infrastructure

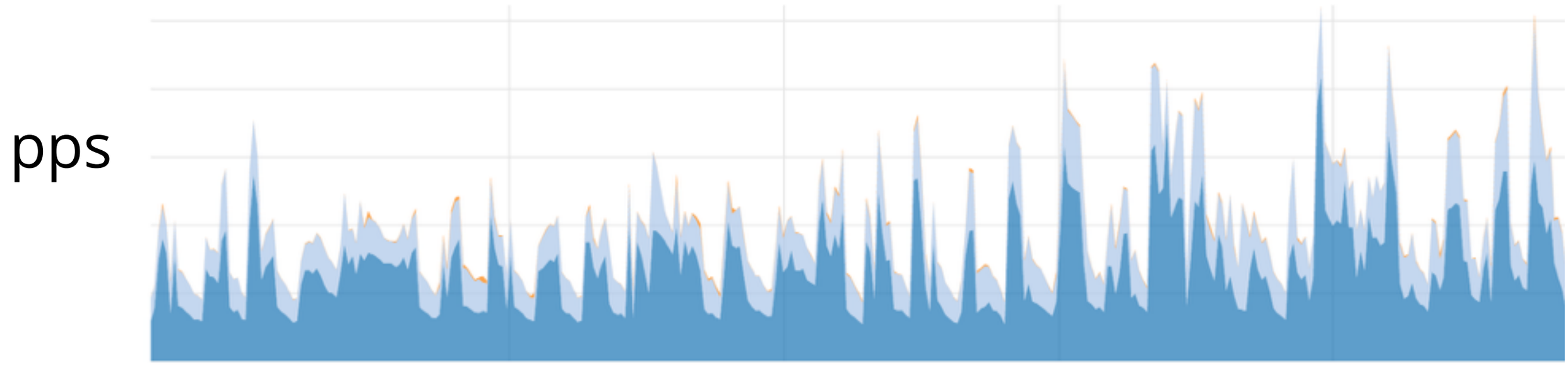
Accuracy takes time

upstream	congestion	more ports, null, topology	ip
router	10M pps	flowspec	ip, proto, length,
network card	6M pps	floodgate	full payload
kernel	1.2M pps	iptables	full payload
DNS server	0.3M pps	selective drops, just handle	full payload

Tools development timeline

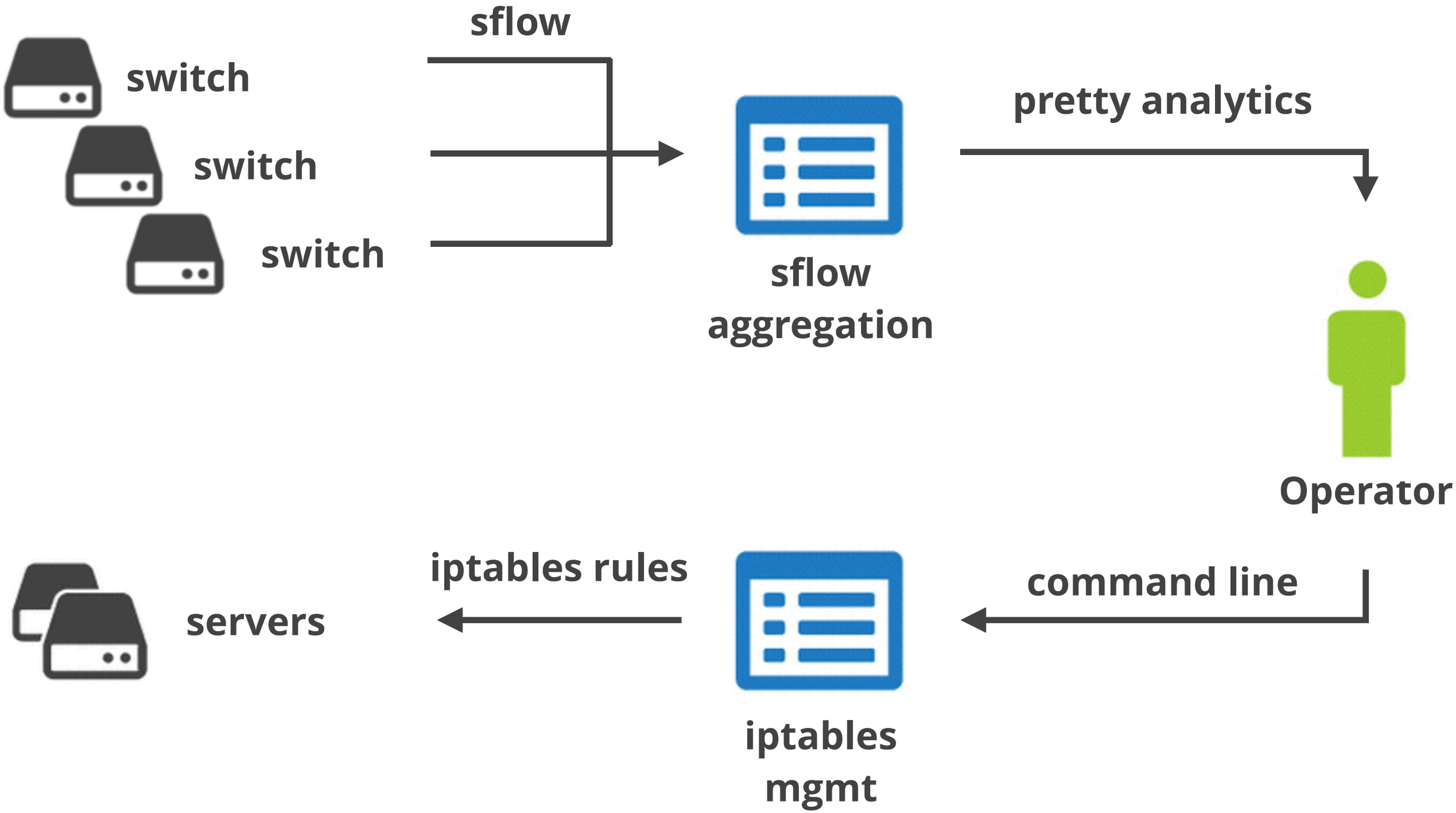


The pain is increasing



30 days →

Manual attack handling



Sflow analytics

```
INCOMING/UDP  samples=11836 pps={total= 65.176M, accounted= 65.174M}
   Mpps   rule hint
  1.160     dnsbpf -- --ip=173.245.59.                www.          com
  3.953     dnsbpf -- --ip=173.245.59.                www.        .com
  3.892     dnsbpf -- --ip=173.245.58.                 .com
  4.169     dnsbpf -- --ip=173.245.58.                 .com
  3.750     dnsbpf -- --ip=173.245.58.                www.        .com
  4.098     dnsbpf -- --ip=173.245.59.                 .com
  0.965     dnsbpf -- --ip=173.245.59.                 .com
  3.910     dnsbpf -- --ip=173.245.59.                  com
  0.973     dnsbpf -- --ip=173.245.59.                www.        .com
  3.811     dnsbpf -- --ip=173.245.58.                www.        .com
  3.770     dnsbpf -- --ip=173.245.59.                 .com
  3.879     dnsbpf -- --ip=173.245.58.                 .com
  3.857     dnsbpf -- --ip=173.245.58.                 .com
  3.975     dnsbpf -- --ip=173.245.58.                www.        .com
  0.545     dnsbpf -- --ip=                .
  3.406     dnsbpf -- --ip=                .
  0.315     dnsbpf -- --ip=173.245.58.
  0.415     dnsbpf -- --ip=173.245.59.                *.          .com
```

Iptables management

```
$ gatekeeper dnsbpf list
[ ] Imported: 0 removed: 0 trusted: 25
--expiry=365d -- --ip=173.245.[redacted] [redacted] www.[redacted].com
--expiry=7d -- --ip=[redacted] m.[redacted].com
--expiry=7d -- --ip=[redacted] m.[redacted].com
--expiry=7d -- --ip=[redacted] www.[redacted].com
--expiry=7d -- --ip=[redacted] www.[redacted].com
--ip=173.245.[redacted] [redacted] www.[redacted].com
--ip=173.245.[redacted] *.www.[redacted].com
--ip=173.245.[redacted] [redacted] www.[redacted].com
--ip=[redacted] [redacted].com
--ip=[redacted] [redacted].com
--ip=[redacted] *.[redacted].com *.[redacted].com *.www.[redacted].com
--ip=[redacted] *.www.[redacted].com *.[redacted].com
--ip=[redacted] *.www.[redacted].com *.www.[redacted].com
--ip=[redacted] [redacted] [redacted], [redacted],
--ip=[redacted] *.[redacted].com *.[redacted].com *.www.[redacted].com
--ip=[redacted] *.[redacted].com *.[redacted].com
```

David Ulevitch

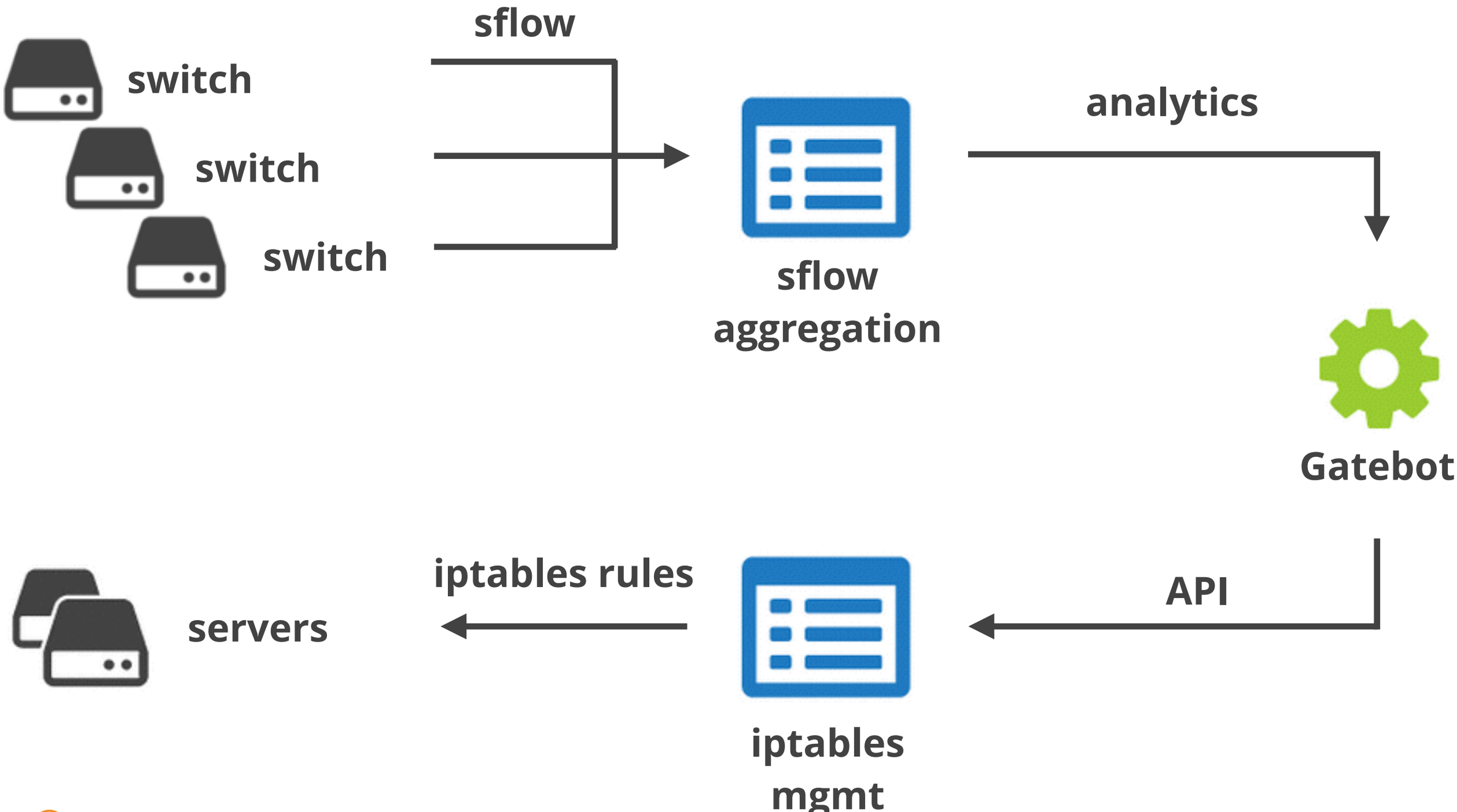
@davidu

Follow

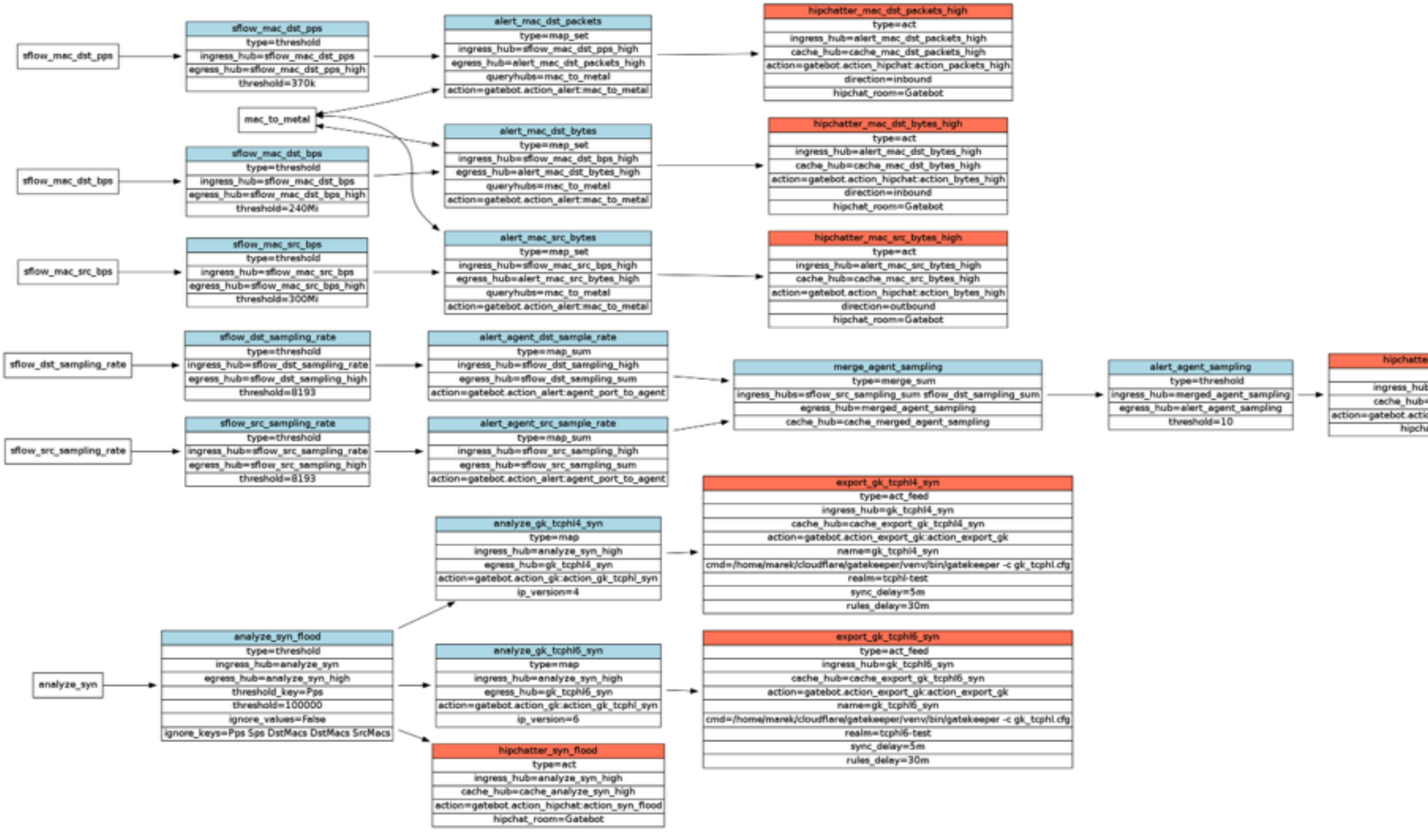
OH nerd trash talk: "The most hyped security startup of 2014 turned out to just be building a GUI for dynamic iptables-based firewalls."

San Francisco, CA

Automatic attack handling

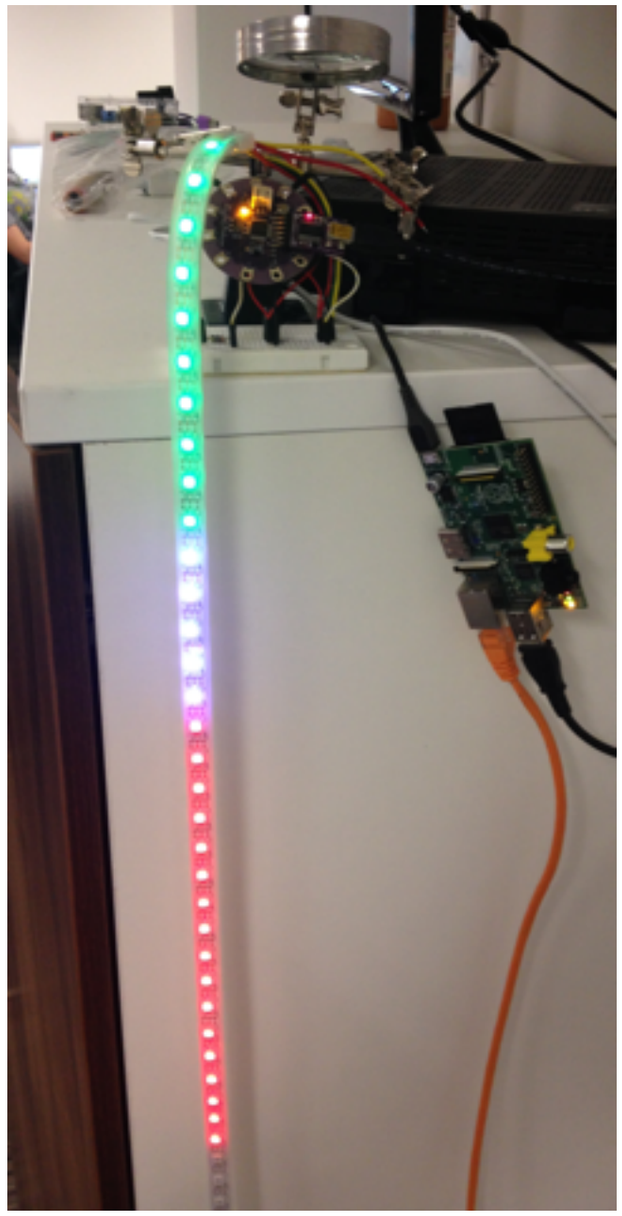


Gatebot


sflow_mac_dst_pps
sflow_mac_dst_pps
type=threshold
ingress_hub=sflow_mac_dst_pps
egress_hub=sflow_mac_dst_pps_high
threshold=370k
alert_mac_dst_packets
type=map_set
ingress_hub=sflow_mac_dst_pps_high
egress_hub=alert_mac_dst_packets_high
queryhubs=mac_to_metal
action=gatebot.action_alert:mac_to_metal
hipchatter_mac_dst_packets_high
type=act
ingress_hub=alert_mac_dst_packets_high
cache_hub=cache_mac_dst_packets_high
action=gatebot.action_hipchat:action_packets_high
direction=inbound
hipchat_room=Gatebot
mac_to_metal
sflow_mac_dst_bps
sflow_mac_dst_bps
type=threshold
ingress_hub=sflow_mac_dst_bps
egress_hub=sflow_mac_dst_bps_high
threshold=240Mi
alert_mac_dst_bytes
type=map_set
ingress_hub=sflow_mac_dst_bps_high
egress_hub=alert_mac_dst_bytes_high
queryhubs=mac_to_metal
action=gatebot.action_alert:mac_to_metal
hipchatter_mac_dst_bytes_high
type=act
ingress_hub=alert_mac_dst_bytes_high
cache_hub=cache_mac_dst_bytes_high
action=gatebot.action_hipchat:action_bytes_high
direction=inbound
hipchat_room=Gatebot
sflow_mac_src_bps
sflow_mac_src_bps
type=threshold
ingress_hub=sflow_mac_src_bps
egress_hub=sflow_mac_src_bps_high
threshold=300Mi
alert_mac_src_bytes
type=map_set
ingress_hub=sflow_mac_src_bps_high
egress_hub=alert_mac_src_bytes_high
queryhubs=mac_to_metal
action=gatebot.action_alert:mac_to_metal
hipchatter_mac_src_bytes_high
type=act
ingress_hub=alert_mac_src_bytes_high
cache_hub=cache_mac_src_bytes_high
action=gatebot.action_hipchat:action_bytes_high
direction=outbound
hipchat_room=Gatebot
sflow_dst_sampling_rate
sflow_dst_sampling_rate
type=threshold
ingress_hub=sflow_dst_sampling_rate
egress_hub=sflow_dst_sampling_high
threshold=8193
alert_agent_dst_sample_rate
type=map_sum
ingress_hub=sflow_dst_sampling_high
egress_hub=sflow_dst_sampling_sum
action=gatebot.action_alert:agent_port_to_agent
merge_agent_sampling
type=merge_sum
ingress_hubs=sflow_src_sampling_sum sflow_dst_sampling_sum
egress_hub=merged_agent_sampling
cache_hub=cache_merged_agent_sampling
alert_agent_sampling
type=threshold
ingress_hub=merged_agent_sampling
egress_hub=alert_agent_sampling
threshold=10
sflow_src_sampling_rate
sflow_src_sampling_rate
type=threshold
ingress_hub=sflow_src_sampling_rate
egress_hub=sflow_src_sampling_high
threshold=8193
alert_agent_src_sample_rate
type=map_sum
ingress_hub=sflow_src_sampling_high
egress_hub=sflow_src_sampling_sum
action=gatebot.action_alert:agent_port_to_agent
export_gk_tcphl4_syn
type=act_feed
ingress_hub=gk_tcphl4_syn
cache_hub=cache_export_gk_tcphl4_syn
action=gatebot.action_export_gk:action_export_gk
name=gk_tcphl4_syn
cmd=/home/marek/cloudflare/gatekeeper/venv/bin/gatekeeper -c gk_tcphl.cfg
realm=tcphl-test
sync_delay=5m
rules_delay=30m
analyze_gk_tcphl4_syn
type=map
ingress_hub=analyze_syn_high
egress_hub=gk_tcphl4_syn
action=gatebot.action_gk:action_gk_tcphl_syn
ip_version=4
analyze_syn
analyze_syn_flood
type=threshold
ingress_hub=analyze_syn
egress_hub=analyze_syn_high
threshold_key=Pps
threshold=100000
ignore_values=False
ignore_keys=Pps Sps DstMacs DstMacs SrcMacs
analyze_gk_tcphl6_syn
type=map
ingress_hub=analyze_syn_high
egress_hub=gk_tcphl6_syn
action=gatebot.action_gk:action_gk_tcphl_syn
ip_version=6
export_gk_tcphl6_syn
type=act_feed
ingress_hub=gk_tcphl6_syn
cache_hub=cache_export_gk_tcphl6_syn
action=gatebot.action_export_gk:action_export_gk
name=gk_tcphl6_syn
cmd=/home/marek/cloudflare/gatekeeper/venv/bin/gatekeeper -c gk_tcphl.cfg
realm=tcphl6-test
sync_delay=5m
rules_delay=30m
hipchatter_syn_flood
type=act
ingress_hub=analyze_syn_high
cache_hub=cache_analyze_syn_high
action=gatebot.action_hipchat:action_syn_flood
hipchat_room=Gatebot

Summary

- Time to mitigation is critical
- Want to be as selective as possible
- Automation is a process, not a project



Thanks

and good luck!

marek@cloudflare.com